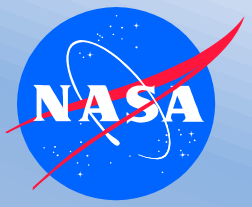
National Aeronautics
and Space Administration



Snapdragon 820 SEE Testing and On-Site Stuck Bit Mitigation

Steven M. Guertin, Sergeh Vartanian, and Matthew Cui

Jet Propulsion Laboratory / California Institute of Technology
Pasadena, CA

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology,
Under contract with the National Aeronautics and Space Administration (NASA)

This work was sponsored by the NASA Electronic Parts and Packaging (NEPP) program. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

National Aeronautics
and Space Administration

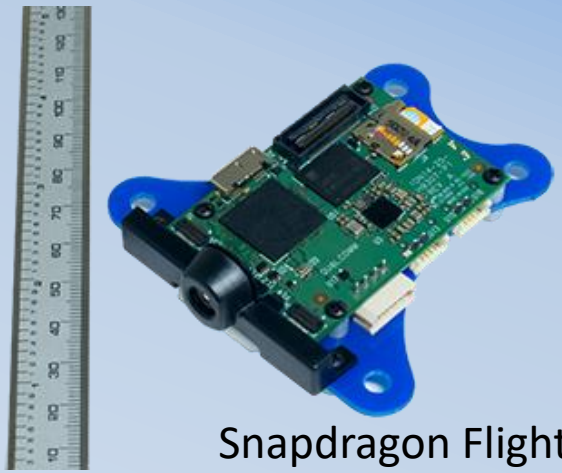
Outline

- Background and Motivation
- Test Setup
- Testing – What happened? & on-site issues
- Problem Solving
- Results
- Future Work
- Conclusions

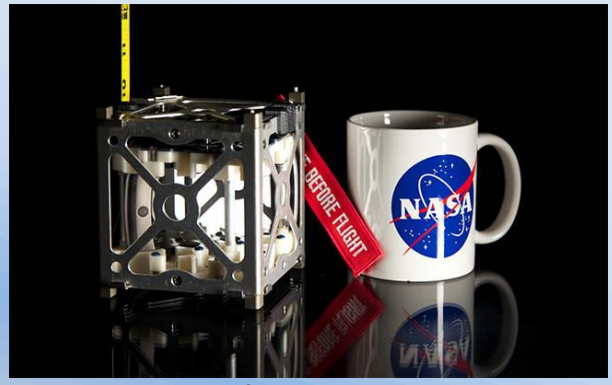
National Aeronautics
and Space Administration

Motivation

- Device built on Samsung 14 nm Low Power Plus (LPP) FinFET platform – fabrication info
- Very low power system (<5 W)
- Vehicle for exploring ARM/Snapdragon architecture



Snapdragon Flight
- Qualcomm



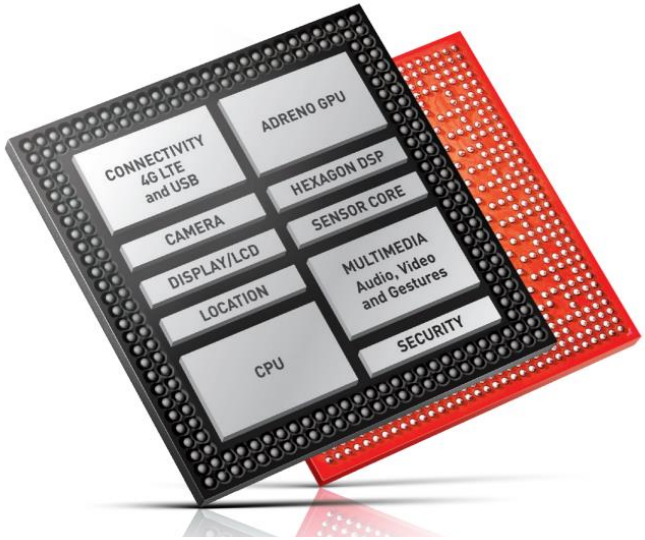
phonesat 2.5

image – vr-zone.com

National Aeronautics and Space Administration

Background: Snapdragon 820

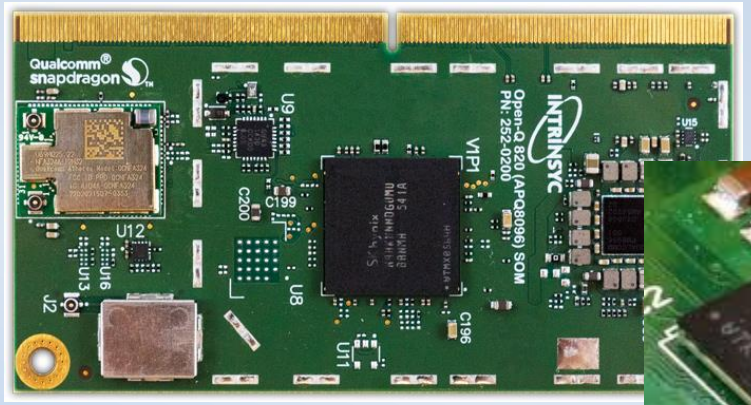
- Key Features:
  - Quad-core Kyro CPU
    - Actually has ~9 distinct processors
    - big.LITTLE – 2 cores are faster, bigger, other two are smaller and slower
  - Low power DDR4
  - Universal Flash Storage
  - Hexagon 680 DSP with isolated sensor power
  - Camera controller
  - Hardware multimedia encode & decode



National Aeronautics and Space Administration

Background: Intrinsyc Open-Q 820

- Evaluation board for Snapdragon 820
- Hardware debug intentionally limited
- Uses system-on-module/carrier configuration
- 3GB DDR4 with POP setup



National Aeronautics and Space Administration

Background: Tests Performed

- Heavy Ions @ TAMU
  - Ion selection range limited…
  - Android & custom code

Beam	LET (MeV-cm^2/mg)	Exposure (cm^2)
N	1	1.2E+07
Ne	6	1.2E+04
Ar	15	4.1E+04

- Protons @ MGH
  - ~1x10^{10} /cm^2 with 100, and 200 MeV, 5x10^9/cm^2 with 50 MeV
  - Android & custom code
- Neutrons at LANSCE
  - ~1x10^{11} /cm^2 with sea level neutron spectrum

National Aeronautics and Space Administration

Heavy Ion Setup - Device Stack Estimate

Item	Matrial	Range of Thicknesses Low	High
Beam Port	Aramica	25.4 µm	
Air	Air	3 cm	
DDR3 Top Plastic	Plastic	100 µm (1.18 g/cm^3)	300 µm (1.85 g/cm^3)
DDR3 Die	Silicon	250 µm	400 µm
DDR3 Metalization	Aluminum	40 µm	60 µm
Air Gap	Air	100 µm (1.18 g/cm^3)	300 µm (1.85 g/cm^3)
Snapdragon Top Plastic	Plastic	100 µm	300 µm
Snapdragon Die	Silicon	500 µm	800 µm

	LET (MeV-cm^2/mg) Minimum	Maximum	Range (µm Si) Maximum	Minimum
N-40 MeV	0.74	1.2	1380	448
Ne-40 MeV	1.8	9	701	Out of Range
Ar-40 MeV	10	20	123	Out of Range

- Using estimated thicknesses to get a range of estimated LETs
- Heavy ion testing (thus far) is general info, so it was most important to show chance of reaching sensitive region

National Aeronautics and Space Administration

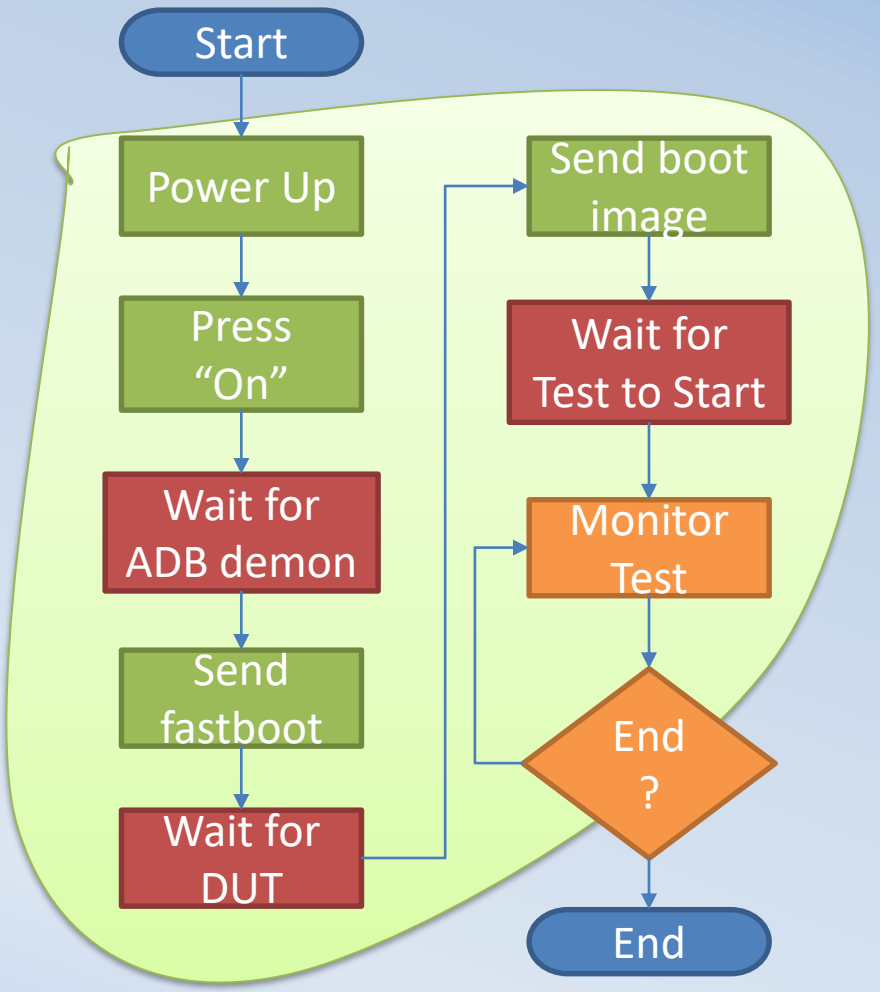
Software Development

- We used Android to initialize the test boards
  - Some testing put the test code before Android, but the data was difficult to work with
  - Later, test code was moved to after Android had brought up I/O
  - During neutron testing we disabled additional cores once we realized we were getting crashes from them
- Software approach is to inject code into boot loader
  - Done by modifying boot binary file directly
- Specific test software items:
  - Memory tests, with L1 cache enabled, @16kB, 64kB, 256kB, and 1GB
  - Let Android run

National Aeronautics
and Space Administration

Test Boot/Run

- Booting was somewhat tricky
- Wanted to avoid bricking the boards, so we never changed the on-board software
- 8 Test System States



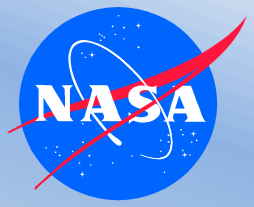
National Aeronautics and Space Administration

Observation: Crashes

- Every operating condition had crashes
  - Mostly these involved the test DUT no longer communicating
  - On later tests, we were able to use Android's exception handlers to get some indication what was going on
- Required restarting the test after each crash
  - We developed an automated system to do this at LANSCE
  - System conditions/handling was complex for the 8 states and possible errors coming from each
  - At LANSCE this all had to be done while being irradiated

National Aeronautics
and Space Administration

Handling Crashes



- Once we were able to isolate cause of crashes
  - This occurred during testing at LANSCE where we could observe the exception messages
  - Also, we were able to tell the exception handlers to return to the test code.
- Most common exceptions were observed to come from CPUs that were not being tested
  - Instructed Android to not boot unnecessary CPUs
- Remaining exceptions were caused primarily by memory access errors

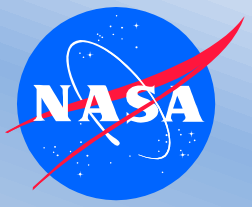
National Aeronautics and Space Administration

Observation: Stuck Bits

- We targeted the DDR4 device for data, because it had to be exposed regardless
  - This is frustrating because it really is not helpful to have two sources of errors
  - But the DDR4 device provided stuck bits as well as SBUs
- Proton and neutron testing → about as many stuck bits as SBUs
  - Stuck bits caused the DUTs to have trouble booting
  - Usually a reasonable chance to get a handful of detectable stuck bits before a DUT was unable to boot
  - Android appears to have a retry option on some memory allocation to allow it to avoid bad memory regions

National Aeronautics
and Space Administration

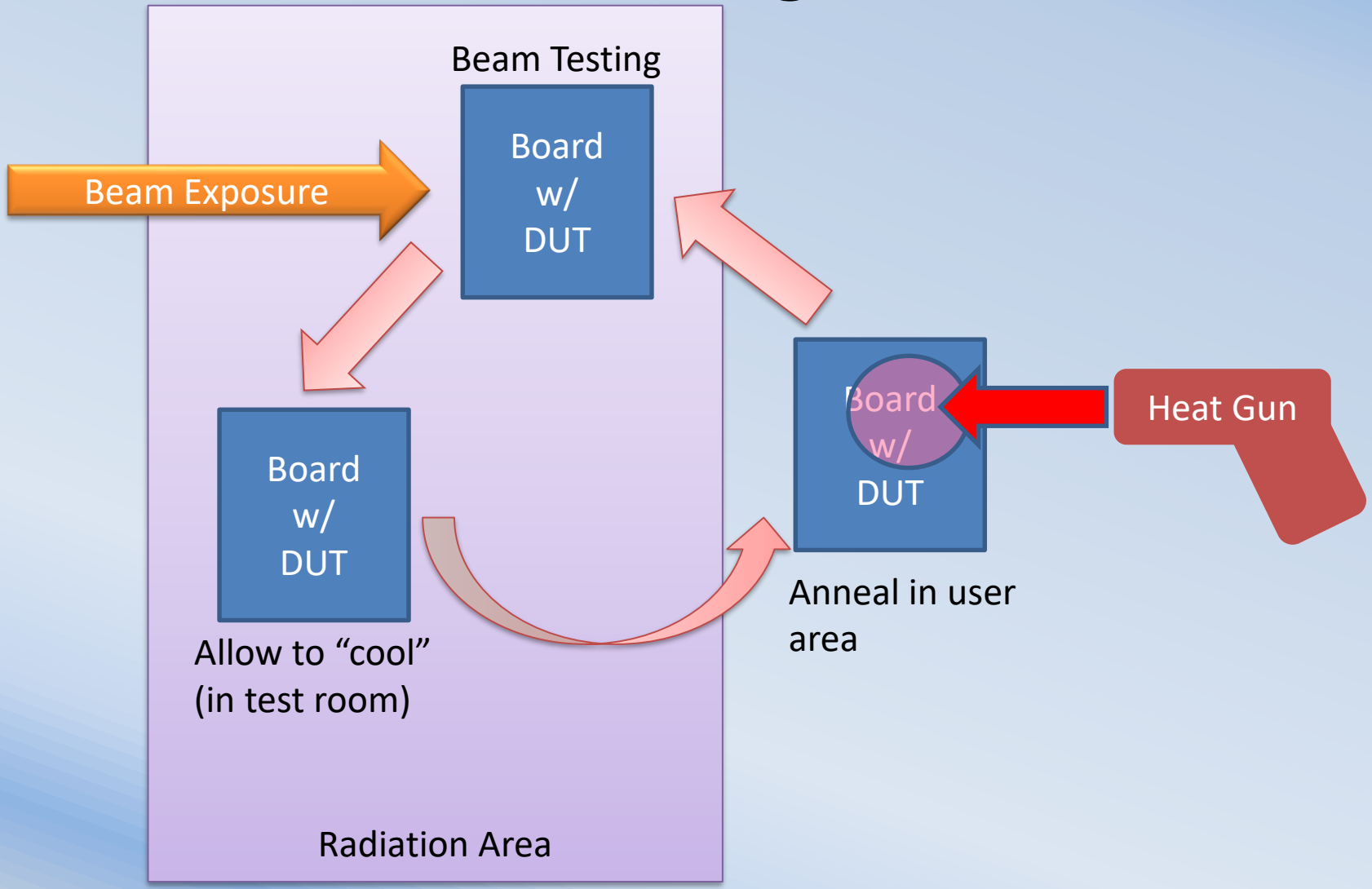
Handling Stuck Bits



- Stuck bits were observed once we were able to get the test system reading 1 GB (out of 3 GB) of the device memory.
- Were able to determine that earlier failures of DUTs were likely due to stuck bits interfering with boot processes.
- Developed methods to anneal stuck bits in-situ, requiring a rotation of boards at LANSCE to facilitate automated testing.

National Aeronautics and Space Administration

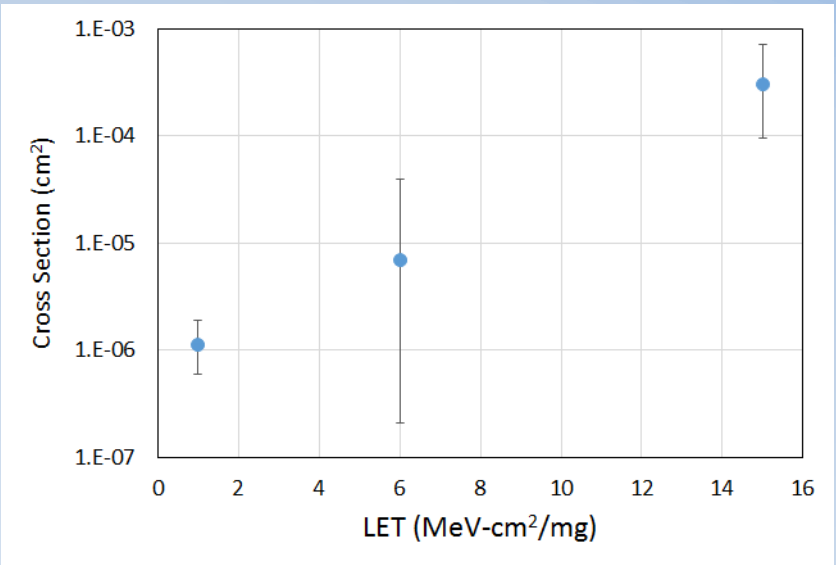
Stuck Bit Annealing



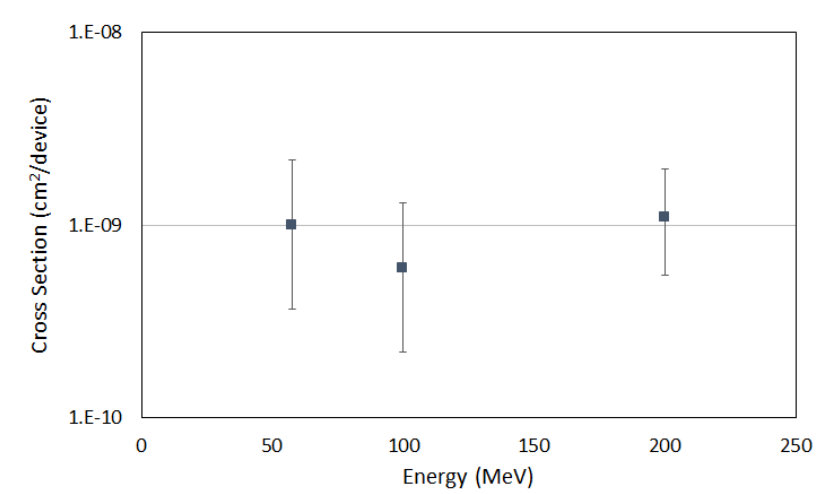
National Aeronautics and Space Administration

Results: Crashes

- Heavy Ions:



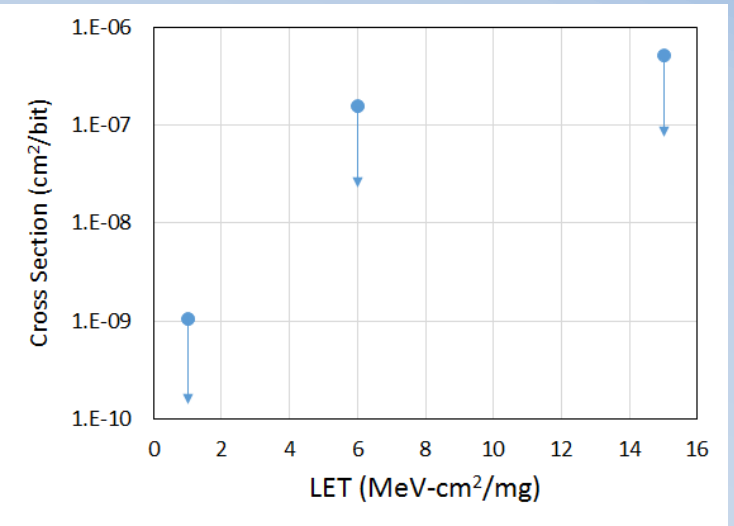
- Protons/Neutrons:
  - Proton curve →
  - Neutrons
    $\sigma \sim 1x10^{-8}/cm^2$



National Aeronautics and Space Administration

Results: SBUs & Stuck Bits

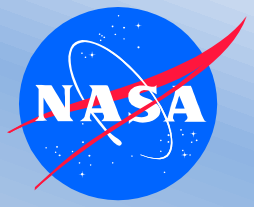
- Limiting σ for SBUs in Snapdragon:



- Stuck Bits during Boot & Anneal:

Board	Incr. Neutron Exposure (/cm2)	Annealing Duration (mins)	Annealing Temp (C)	Stuck Bits - boot Before	Stuck Bits - boot After
2	2.52E+10	30	175	13	1
1	2.52E+10	15	175	14	3
1	0.00E+00	90	175	3	0
4	3.12E+09	120	175	3	0
5	2.81E+10	220	175	8	0

- Note also crunching data on bit errors in the 1GB memory region



Future Work
Heavy Ion Testing

- Improved test system will enable testing with heavy ions
  - Registers
  - Crashes (with capture of exceptions)
  - DDR4 Errors
- Additional data on caches
  - We have modified cache test code, but so far the L1 caches have not shown bit errors
    - May have ECC or parity masking the errors
- Test code on all cores?

National Aeronautics
and Space Administration

Conclusion

- Snapdragon 820 test development
  - How to deal with package on package – blast through it
  - Software – able to put custom low-level software in beam
- System results
  - Primary events observed – crashes, and bit errors in DDR4
  - Stuck bits also observed in DDR4
    - Sometimes Android was able to map around these during boot
  - No cache bit errors observed
- In-situ test development to improve data
  - On-site annealing of stuck bits used to recover boot
  - Automated exception responses to enable continuing test